Examiner-Initiated Interview Summary	**Application No.** 10/654,743	**Applicant(s)** HEN ET AL.	
	Examiner KimbleAnn Verdi	**Art Unit** 2194	

All Participants:

Status of Application: *now Allowed*

(1) *KimbleAnn Verdi*.

(2) *David W. Victor*.

(3) _____.

(4) _____.

Date of Interview: *25 February 2008*

Time: *11:30 am*

Type of Interview:

☒ Telephonic
☐ Video Conference
☐ Personal (Copy given to: ☐ Applicant ☐ Applicant's representative)

Exhibit Shown or Demonstrated: ☐ Yes ☒ No
If Yes, provide a brief description: .

Part I.

Rejection(s) discussed:
n/a

Claims discussed:
1-5, 8-9, 11-19, 22, 24-31, 34, 36-39.

Prior art documents discussed:
n/a

Part II.

SUBSTANCE OF INTERVIEW DESCRIBING THE GENERAL NATURE OF WHAT WAS DISCUSSED:

Applicant agreed to ammend the claims 1-5, 8-9, 11-19, 22, 24-31, 34, 36-39 as presented in the examiner's amendment. Applicant authorized the ammendment to be made in an Examiner's Ammendment.

Part III.

☒ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview directly resulted in the allowance of the application. The examiner will provide a written summary of the substance of the interview in the Notice of Allowability.

☐ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview did not result in resolution of all issues. A brief summary by the examiner appears in Part II above.

WILLIAM THOMSON
SUPERVISORY PATENT EXAM

(Applicant/Applicant's Representative Signature – if appropriate)

KONRAD RAYNES & VICTOR, LLP

315 S. Beverly Drive, Suite 210
Beverly Hills, California 90212

Telephone: (310) 556-7983
Facsimile: (310) 556-7984

FAX COVER SHEET

PLEASE DELIVER THIS FACSIMILE TO EXAMINER KACY VERDI

TO:	Commissioner for Patents	FROM:	David W. Victor
	Attn: Examiner Kacy Verdi	OUR REF:	0077.0019
	Group Art Unit: 2194	TELEPHONE:	310-556-7983

Total pages, including cover letter: **11**

PTO FAX NUMBER: 1-571-270-2654

If you do NOT receive all of the pages, please telephone us at 310-556-7983, or fax us at 310-556-7984.

Description of Documents Transmitted: **SUPPLEMENT PROPOSED AMENDMENTS**

Applicant:	S. HEN et al.
Serial No.:	10/654,743
Filed:	September 3, 2003
Group Art Unit:	2194
Docket No.:	P16176

CERTIFICATE UNDER 37 CFR 1.8:
I hereby certify that this correspondence is being transmitted by facsimile to Examiner Kacy Verdi of the U.S. Patent and Trademark Office on February 29, 2008.

/David Victor/
David W. Victor

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	S. HEN et al.	Examiner	Kimbleann Verdi
Serial No.	10/654,743	Group Art Unit	2194
Filed	September 3, 2003	Docket No.	P16176
TITLE	METHOD, SYSTEM, AND PROGRAM FOR ACCESSING DEVICE DRIVER FUNCTIONS		

CERTIFICATE UNDER 37 CFR 1.8:
I hereby certify that this correspondence is being transmitted through the USPTO EFS-Web system over the Internet to Kimbleann Verdi of the U.S. Patent and Trademark Office on February 29, 2008.

/David Victor/
David W. Victor

SUPPLEMENTAL PROPOSED AMENDMENT TO THE CLAIMS

Applicants submit herein proposed amendments to the claims the Examiner suggested on February 25, 2008.

Applicants amended claim 27 to limit the article of manufacture to computer readable storage medium as disclosed in para. 22 of the Specification and not cover transmission signals alone.

Applicants authorize the Examiner to enter these proposed amendments. Should any additional fees be required beyond those paid, please charge Deposit Account No. 50-0585.

Proposed Amendments to the Claims are reflected in the listing of claims which begins on page 1.

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (Currently Amended) A method for communicating with a device, comprising:

executing a kernel module in a memory;

executing at least one kernel thread in the memory to execute device driver functions for the kernel module, wherein the device driver functions are capable of being invoked by system calls from applications operating in a user context; and

executing, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context;

buffering a parameter list;

accessing, with one of the at least one kernel thread, device information from the device;

setting device ~~parameters~~ values in the buffered parameter list to the accessed device information from the device ~~values provided by kernel module functions~~;

setting a flag indicating that the at least one kernel thread needs to set parameters at the device to device parameter values set in the buffered parameter list

processing, by one of the at least one ~~executed~~ kernel thread, the buffered parameter list in response to the flag being set by performing:

applying a lock on information in the buffered parameter list including the ~~located buffered~~ device parameter values;

after applying the lock, copying the device parameter values from the buffered parameter list to a temporary buffer, wherein ~~the device~~ parameters at the device are set to the device parameter values from the buffered parameter list in the temporary buffer; and

releasing the lock after copying the device parameter values from the buffered parameter list to the temporary buffer.

2. (Original) The method of claim 1, wherein the kernel module spawns the at least one kernel thread to execute the calls to the device driver functions for the kernel module.

3. (Canceled)

4. (Currently Amended) The method of claim [[3]] 1, wherein a kernel module function requests device information, further comprising:

in response to the request for the device information, accessing the buffered device information.

5. (Currently Amended) The method of claim 1, wherein the at least one kernel thread accesses buffered device information periodically and independently of kernel module requests for the device information.

6. (Canceled)

7. (Canceled)

8. (Previously Presented) The method of claim 1, further comprising:

spawning a kernel thread to set device parameters to parameter values buffered in the parameter list.

9. (Currently Amended) The method of claim [[1]] 8, wherein the kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

10. (Canceled)

11. (Previously Presented) The method of claim 1, further comprising:

disabling higher priority contexts before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

12. (Original) The method of claim 11, wherein the higher priority context comprises a bottom half or Interrupt Request (IRQ) context.

13. (Previously Presented) The method of claim 1, further comprising:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

14. (Original) The method of claim 1, further comprising:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.

15. (Currently Amended) A system, comprising:

a network device;

a memory;

a processor executing code to perform:

execute a network device driver in memory to control the network device;

execute a kernel module in the memory;

execute at least one kernel thread in the memory to execute device driver functions for the kernel module, wherein the device driver functions are capable of being invoked by system calls from applications operating in a user context; and

execute, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context;

buffering a parameter list;

accessing, with one of the at least one kernel thread, device information from the device;

setting device ~~parameters~~ values in the buffered parameter list to the accessed device information from the device ~~values provided by kernel module functions~~;

setting a flag indicating that the kernel thread needs to set parameters at the device to device parameter values set in the buffered parameter list;

processing, by one of the at least one ~~executed~~ kernel thread, the buffered parameter list in response to the flag being set by performing:

applying a lock on information in the buffered parameter list including ~~the located buffered~~ device parameter values;

after applying the lock, copying the device parameter values from the buffered parameter list to a temporary buffer, wherein ~~the device~~ parameters at the device are set to the parameter values from the buffered parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the buffered parameter list to the temporary buffer.

16. (Currently Amended) The system of claim 15, wherein the kernel module spawns the at least one kernel thread to execute the called device driver functions.

17. (Canceled)

18. (Currently Amended) The system of claim [[17]] 15, wherein a kernel module function requests device information, wherein the processor executes the code to further perform:

in response to the request for the device information, accessing the buffered device information.

19. (Currently Amended) The system of claim 15, wherein the the at least one kernel thread accesses device information periodically and independently of kernel module requests for device information.

20. (Canceled)

21. (Canceled)

22. (Currently Amended) The system of claim 15, wherein the at least one kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

23. (Canceled)

24. (Previously Presented) The system of claim 15, wherein the processor executes the code to further perform:

disabling higher priority context before locking the parameter list; and
enabling the higher priority contexts after releasing the lock on the parameter list.

25. (Previously Presented) The system of claim 15, wherein the processor executes the code to further perform:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

26. (Original) The system of claim 15, wherein the processor executes the code to further perform:

initiating, with the kernel module, an access request with respect to device information;
disabling any higher priority contexts capable of accessing the device information;
obtaining a lock for the device information subject to the access request;
providing the kernel module access to the device information;
releasing the lock; and
enabling all higher priority contexts that were disabled.

27. (Currently Amended) An article of manufacture comprising a computer readable storage medium having code executed by a processor for communicating with a device and to perform operations, ~~wherein the article of manufacture causes operations to be performed,~~ the operations comprising:

executing a kernel module;

executing at least one kernel thread to execute device driver functions for the kernel module, wherein the device driver functions are capable of being invoked system calls from applications operating in a user context;

executing, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context;

buffering a parameter list;

accessing, with one of the at least one kernel thread, device information from the device;

setting device ~~parameters~~ values in the buffered parameter list to the accessed device information from the device ~~to values provided by kernel module functions~~;

setting a flag indicating that the at least one kernel thread needs to set parameters at the device to device parameter values set in the buffered parameter list; and

processing, by one of the at least one kernel thread, processes the buffered parameter list in response to the flag being set by performing:

applying a lock on information in the buffered parameter list including the ~~located buffered~~ device parameter values;

after applying the lock, copying the device parameter values from the buffered parameter list to a temporary buffer, wherein ~~the device~~ parameters at the device are set to the device parameter values from the buffered parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the buffered parameter list to the temporary buffer.

28. (Original) The article of manufacture of claim 27, wherein the kernel module spawns at least one kernel thread to execute the called device driver functions.

29. (Canceled)

30. (Currently Amended) The article of manufacture of claim [[29]] 27, wherein a kernel module function requests device information, wherein the operations further comprise:

in response to a request for the device information, accessing the buffered device information.

31. (Currently Amended) The article of manufacture of claim 27, wherein the at least one kernel thread accesses buffered device information periodically and independently of kernel module requests for device information.

32. (Canceled)

33. (Canceled)

34. (Currently Amended) The article of manufacture of claim 27, wherein the at least one kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

35. (Canceled)

36. (Previously Presented) The article of manufacture of claim 27, wherein the operations further comprise:

disabling higher priority contexts before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

37. (Original) The article of manufacture of claim 36, wherein the higher priority context comprises a bottom half or Interrupt Request (IRQ) context.

38. (Previously Presented) The article of manufacture of claim 27, wherein the operations further comprise:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

39. (Original) The article of manufacture of claim 27, wherein the code executes operations to further perform:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.

The attorney of record invites the Examiner to contact him at (310) 553-7977 if the Examiner believes such contact would advance the prosecution of the case.

Dated: February 29, 2008

By: /David Victor/

David W. Victor
Registration No. 39,867

Please direct all correspondences to:

David W. Victor
Konrad Raynes & Victor, LLP
315 South Beverly Drive, Ste. 210
Beverly Hills, CA 90212
Tel: (310) 553-7977
Fax: 310-556-7984